# Understanding Ultra-Scale Application Communication Requirements

Shoaib Kamil, John Shalf, Leonid Oliker, David Skinner
*CRD/NERSC, Lawrence Berkeley National Laboratory, Berkeley, CA 94720*

## ABSTRACT

As thermal constraints reduce the pace of CPU performance improvements, the cost and scalability of future HPC architectures will be increasingly dominated by the interconnect. In this work we perform an in-depth study of the communication requirements across a broad spectrum of important scientific applications, whose computational methods include: finite-difference, lattice-bolzmann, particle in cell, sparse linear algebra, particle mesh ewald, and FFT-based solvers. We use the IPM (integrated Performance Monitoring) profiling framework to collect detailed statistics on communication topology and message volume with minimal impact to code performance. By characterizing the parallelism and communication requirements of such a diverse set of applications, we hope to guide architectural choices for the design and implementation of interconnects for future HPC systems.

## 1. INTRODUCTION

As the field of scientific computing matures, the demands for computational resources are growing at a rapid rate. It is estimated that by the end of this decade, numerous mission-critical applications will have computational requirements that are at least two orders of magnitude larger than current levels [1]. However, as the pace of processor clock rate improvements continues to slow, the path towards realizing peta-scale computing is increasingly dependent on scaling up the number of processors to unprecedented levels.

As a result, there is a critical need to understand the range of CPU interconnectivity required by parallel scientific codes. The massive parallelism required for ultra-scale computing presents two competing risks. If interconnects do not provide sufficient connectivity and data rates for the communication inherent in HPC algorithms, application performance will not scale. If interconnects are over-built, however, they will dominate the overall cost and thus the extent of ultra-scale systems. Both performance scalability and cost scalability are crucial to manufacturers and users of large scale parallel computers. Failure to characterize and understand the communication requirements of HPC codes risks limiting the extent of future scientific computation.

In this work, we begin by examining the current state of HPC interconnects in the next section, motivating some of the characteristics we examine. Then we outline our profiling methodology, introducing the IPM library as well as the applications we study. We then explore the observed characteristics of our applications, including buffer sizes and connectivity, ending with an analysis of the feasibility of lower-degree interconnects for ultra-scale computing.


*IISWC* 2005, Austin, TX, USA

## 2. INTERCONNECTS FOR HPC

HPC systems implementing *fully-connected networks* (FCNs) such as fat-trees and crossbars have proven popular due to their excellent bisection bandwidth and ease of application mapping for arbitrary communication topologies. However, it is becoming increasingly difficult and expensive to maintain these types of interconnects, since the cost of an FCN infrastructure composed of packet switches grows superlinearly with the number of nodes in the system. As supercomputing systems with tens or even hundreds of thousands of processors begin to emerge, FCNs will quickly become infeasibly expensive. Recent studies of application communication requirements, such as the series of papers from Vetter and Mueller [18, 19], have observed that applications with the best scaling efficiency have communication topology requirements that are far less than the total connectivity provided by FCN networks.

Concerns about the cost and complexity of interconnection networks on next-generation MPPs has caused a renewed interest in networks with a lower topological degree, such as mesh and torus interconnects (like those used in IBM BlueGene/L, Cray RedStorm, and Cray X1), whose costs rise linearly with system scale. The most significant concern is that lower-degree interconnects may not provide suitable performance for all flavors of scientific algorithms. However, even if the network offers a topological degree that is greater than or equal to the application's *topological degree of connectivity* (TDC), traditional low-degree interconnect approaches have several significant limitations. For example, there is no guarantee the interconnect and application communication topologies are isomorphic—hence preventing the communication graph from being properly embedded into the fixed interconnect topology. Before moving to a radically different interconnect solution, it is essential to understand scientific application communication requirements across a broad spectrum of numerical methods.

In subsequent sections of this work we present a detailed analysis of application communication requirements for a

| System | Technology | MPI Latency | Peak Bandwidth | Bandwidth Delay Product |
|---|---|---|---|---|
| SGI Altix | Numalink-4 | 1.1us | 1.9 GB/s | 2 KB |
| Cray X1 | Cray Custom | 7.3us | 6.3 GB/s | 46 KB |
| NEC Earth Simulator | NEC Custom | 5.6us | 1.5 GB/s | 8.4 KB |
| Pathscale | Infinipath | 1.5us | 1.0 GB/s | 1.5 KB |
| Myrinet Cluster | Myrinet 2000 | 5.7us | 500 MB/s | 2.8 KB |
| Cray XD1 | RapidArray/IB4x | 1.7us | 2 GB/s | 3.4 KB |

**Table 1: Bandwidth delay products for several high performance interconnect technologies. This is the effective peak unidirectional bandwidth delivered per CPU (not per link).**

number of important supercomputing applications and explore the feasibility of lower-degree interconnects for these applications.

## 2.1 Small Messages and the Bandwidth Delay Product

Not every message in a given communication topology is an important message. In particular very small messages do not benefit from a dedicated link because they are unable to saturate the available bandwidth of the network interface. We use the *bandwidth-delay* product to define more precisely the size of messages that would otherwise be unable to take advantage of a dedicated link.

The product of the bandwidth and the delay for a given point-to-point connection describes precisely how many bytes must be "in-flight" to fully utilize available link bandwidth. This can also be thought of as the minimum size required for a non-pipelined message to fully utilize available link bandwidth. Vendors commonly refer to an $N_{1/2}$ metric, which describes the message size below which you will get only 1/2 of the peak link performance. The $N_{1/2}$ metric is typically very close to half the bandwidth-delay product. Table 1 shows the bandwidth delay products for a number of leading-edge interconnect implementations.

When considering point-to-point interconnect bandwidth, it is important to focus on messages that are larger than the bandwidth-delay product. Such messages can derive benefit from bonded or striped interconnect fabrics where bandwidth is built-up incrementally using multiple links that follow the same topology. Messages that are less than $N_{1/2}$ require a focus on latency minimization, since latency-bound messages do not benefit from bonding or striped interconnect fabrics. For this work, we consider messages sizes smaller than 2 KB to be latency-bound, and therefore not as important as larger point-to-point messages. While it is possible to pipeline the messages in order to reach saturation, the overhead incurred by most MPI implementations renders such pipelining ineffective.

# 3. METHODOLOGY

## 3.1 IPM: Low-overhead MPI profiling

The Integrated Performance Monitoring (IPM) [2] application profiling layer allows us to non-invasively gather the communication characteristics of parallel codes as they are run in a production environment. IPM brings together multiple sources of performance metrics into a single profile that characterizes the overall performance and resource usage of the application. It maintains low overhead by using a unique hashing approach, yielding a fixed memory footprint and minimal CPU usage. IPM is open source, relies on portable software technologies, and is scalable to thousands of tasks.

Since most of the workload we are interested in uses MPI for parallelism, we have focused on implementing IPM through the name shifted profiling interface to MPI. The use of the profiling interface to MPI is of widely recognized value in profiling MPI codes [16, 18]. The name-shifted or PMPI interface allows each MPI call to be wrapped by profiling code that collects communication performance information.

IPM collects a wide variety of communication information through this interface. In this work, we are principally using the information that encodes the number of each MPI call with a unique set of arguments. Arguments to MPI calls contain message buffer size as well as source and destination information. In some cases, we also track information from the `MPI_Status` structure. For instance, in the case of `MPI_Send`, IPM keeps track of each unique buffer size and destination, the number of such calls, as well as the total, minimum, and maximum runtimes to complete the call. IPM also allows code regions to be defined, enabling us to separate application initialization from steady state computation and communication patterns, as we are interested, primarily, in the communication topology for the application in its post-initialization steady state. We ran using IPM on Seaborg, the NERSC IBM SP. Since the applications studied here do not use performance-adaptive algorithms, the communication data presented here would be the same on any other interconnect. Overall, IPM captures the topology and nature of communication, which is essential for understanding the applicability of various interconnect architectures to real-world scientific application requirements.

## 3.2 Evaluated Applications

In this section we highlight the salient features of the applications studied in this work. The high level overview of the codes and methods is presented in Table 2. Each of these applications is actively run at multiple supercomputing centers, consuming a sizable amount of computational resources. Detailed descriptions of the algorithms and scientific impact of these codes has been explored elsewhere [6, 7, 8, 11, 12, 13, 15].

### *3.2.1 Cactus*

One of the most challenging problems in astrophysics is the numerical solution of Einstein's equations following from the Theory of General Relativity (GR): a set of coupled nonlinear hyperbolic and elliptic equations containing thousands of terms when fully expanded. The Cactus Computational ToolKit [3] is designed to evolve these equations stably in 3D on supercomputers to simulate astrophysical phenomena with high gravitational fluxes, such as the collision of two black holes and the gravitational waves radiating from that event. The Cactus GR components solve Einstein's equations as an initial value problem that evolves partial differential equations (PDEs) on a regular grid us-

| Name | Lines | Discipline | Problem and Method | Structure |
|---|---|---|---|---|
| Cactus [3] | 84,000 | Astrophysics | Einstein's Theory of GR via Finite Differencing | Grid |
| LBMHD [13] | 1,500 | Plasma Physics | Magneto-Hydrodynamics via Lattice Boltzmann | Lattice/Grid |
| GTC [12] | 5,000 | Magnetic Fusion | Vlasov-Poisson Equation via Particle in Cell | Particle/Grid |
| SuperLU [11] | 42,000 | Linear Algebra | Sparse Solve via LU Decomposition | Sparse Matrix |
| PMEMD | 37,000 | Life Sciences | Molecular Dynamics via Particle Mesh Ewald | Particle |
| PARATEC [8] | 50,000 | Material Science | Density Functional Theory via FFT | Fourier/Grid |
| FVCAM [15] | 200,000 | Climate Modeling | Atmospheric Circulation via Finite Volume | Grid |
| MADbench [6] | 5000 | Cosmology | CMB Analysis via Newton-Raphson | Dense Matrix |

**Table 2: Overview of scientific applications examined in this work.**

ing finite differencing. For parallel computation, the global 3D grid is block domain decomposed so that each processor has its own section. The standard MPI driver for Cactus solves the PDEs on a local region and then updates the values at the ghost zones by exchanging data on the faces of its topological neighbors.

#### 3.2.2 GTC

The Gyrokinetic Toroidal Code (GTC) is a 3D particle-in-cell (PIC) application developed at the Princeton Plasma Physics Laboratory to study turbulent transport in magnetic confinement fusion [12]. GTC solves the non-linear gyrophase-averaged Vlasov-Poisson equations [10] for a system of charged particles in a self-consistent, self-generated electrostatic field. The geometry is that of a torus with an externally imposed equilibrium magnetic field, characteristic of toroidal fusion devices. By using the Particle-in-Cell method, the non-linear PDE describing the motion of the particles in the system becomes a simple set of ordinary differential equations (ODEs) that can be easily solved in Lagrangian coordinates. The self-consistent electrostatic field driving this motion could conceptually be calculated directly from the distance between each pair of particles using an $O(N^2)$ calculation, but the PIC approach reduces it to $O(N)$ by using a grid where each particle deposits its charge to a limited number of neighboring points according to its range of influence.

#### 3.2.3 LBMHD

Lattice Boltzmann methods (LBM) have proved a good alternative to conventional numerical approaches for simulating fluid flows and modeling physics in fluids. The basic idea of the LBM is to develop a simplified kinetic model that incorporates the essential physics, and reproduces correct macroscopic averaged properties. Recently, several groups have applied the LBM to the problem of magneto-hydrodynamics (MHD) with promising results. LBMHD [14] simulates the behavior of a three-dimensional conducting fluid evolving from simple initial conditions and decaying to form current sheets. This application uses either a 2D or 3D cubic lattice for spatial and velocity resolution with 9 or 27 streaming vectors, respectively. Each grid point is associated with a set of mesoscopic variables, whose values are stored in vectors proportional to the number of streaming directions – in this case nine (eight plus the null vector) or 27. The simulation proceeds by a sequence of collision and stream steps. In this paper, we examine both the 2D and 3D cases.

#### 3.2.4 SuperLU

SuperLU [11] is a general purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines. The library is written in C and is callable from either C or Fortran. The library routines perform an LU decomposition with partial pivoting and triangular system solves through forward and back substitution. This application relies on sparse linear algebra for its main computational kernels.

#### 3.2.5 PMEMD

PMEMD (Particle Mesh Ewald Molecular Dynamics) is an application that performs molecular dynamics simulations and minimizations. The force evaluation is performed in an efficiently-parallel manner using state of the art numerical and communication methodologies. PMEMD uses a highly asynchronous approach to communication for the purpose of achieving a high degree of parallelism. Periodic load balancing steps redistribute the spatially decomposed grid amongst MPI tasks.

#### 3.2.6 PARATEC

PARATEC (PARAllel Total Energy Code [8]) performs ab-initio quantum-mechanical total energy calculations using pseudopotentials and a plane wave basis set. PARATEC uses an all-band conjugate gradient (CG) approach to solve the Kohn-Sham equations of Density Functional Theory (DFT) and obtain the ground-state electron wavefunctions. DFT is the most commonly used technique in materials science, having a quantum mechanical treatment of the electrons, to calculate the structural and electronic properties of materials. Codes based on DFT are widely used to study properties such as strength, cohesion, growth, magnetic/optical properties, and transport for materials like nanostructures, complex surfaces, and doped semiconductors. Due to their accurate predictive power and computational efficiency, DFT-based codes have become some of the largest consumers of supercomputing cycles in computer centers around the world. In solving the Kohn-Sham equations using a plane wave basis, part of the calculation is carried out in real space and the remainder in Fourier space using specialized parallel 3D FFTs to transform the wavefunctions.

#### 3.2.7 FVCAM

The Community Atmosphere Model (CAM) is the atmospheric component of the flagship Community Climate System Model (CCSM3.0). Developed at the National Center for Atmospheric Research (NCAR), the CCSM3.0 is extensively used to study climate change. The CAM application is an atmospheric general circulation model (AGCM) and can be run either coupled within CCSM3.0 or in a stand-alone mode driven by prescribed ocean temperatures and sea ice coverages. The dynamic core of CAM was constructed with two very different methodologies to solve the equations of

| Function | Cactus | GTC | LBMHD | PARATEC | PMEMD | SuperLU | FVCAM2D | MADbench |
|---|---|---|---|---|---|---|---|---|
| Isend | 26.8% | 0% | 40% | 25.1% | 32.7% | 16.4% | 28.3% | 5.3% |
| Irecv | 26.8% | 0% | 40% | 24.8% | 29.3% | 15.7% | 28.3% | 0% |
| Wait | 39.3% | 0% | 0% | 49.6% | 0% | 30.6% | 19.8% | 0% |
| Waitall | 6.5% | 0% | 20% | 0.1% | 0.6% | 0% | 22.9% | 0% |
| Waitany | 0% | 0% | 0% | 0% | 36.6% | 0% | 0% | 0% |
| Sendrecv | 0% | 40.8% | 0% | 0% | 0% | 0% | 0% | 30.1% |
| Send | 0% | 0% | 0% | 0% | 0% | 14.7% | 0% | 32.2% |
| Gather | 0% | 47.4% | 0% | 0.02% | 0% | 0% | 0% | 0% |
| Reduce/Allreduce | 0.5% | 11.7% | 0.06% | 0% | 0.7% | 1.9% | 0.5% | 13.6% |
| Bcast | 0% | 0.04% | 0% | 0.03% | 0% | 5.3% | 0% | 6.8% |

**Table 3: Breakdown of MPI communication call percentages for each of the codes.**

motion. The default method, known as the spectral transform method, exploits spherical harmonics to map a solution onto the sphere. An alternate formulation based on a finite volume methodology is also supplied. This option, referred to in this paper as FVCAM, is based on a regular latitude-longitude mesh and conserves certain higher order moments, and thus allows a 2D domain decomposition. In this work we examine the communication characteristics of both CAM and FVCAM.

#### *3.2.8 MADbench*

The Cosmic Microwave Background (CMB) is a snapshot of the Universe when it first became electrically neutral some 400,000 years after the Big Bang. The tiny anisotropies in the temperature and polarization of the CMB radiation are sensitive probes of cosmology. MADbench [6] is a lightweight version of the Microwave Anisotropy Dataset Computational Analysis Package (MADCAP) CMB power spectrum estimation code that retains the operational complexity and integrated system requirements of the original. The computation estimates the angular power spectrum using a Newton-Raphson iteration to locate the peak of the spectral likelihood function, via dense linear algebra operations. MADbench's most computationally intensive component — a set of independent dense matrix-matrix multiplications — was originally performed in single-gang mode, where all the processors processed each matrix one at a time. The recent introduction of gang-parallelism enables more efficient matrix multiplications but requires an additional data remapping transformation. In this paper we refer to single- and gang-parallel implementations as MADbench SG and MADbench MG respectively.

### 3.3 Application Summary

Together, this sampling of applications spans the characteristics of a great many more scientific computing applications, especially with respect to communication pattern. For instance, though we examine PARATEC in the present work, its core algorithm shares the communication characteristics of many other important plane-wave DFT codes (CPMD, VASP, etc.). Likewise, we expect a large number of finite difference and particle-mesh codes to exhibit similar communication patterns, based on our study of Cactus and PMEMD. Certain reduced quantities important to the present study, such as communication degree, should be largely dictated by the problem solved and algorithmic methodology. For instance, in the case of Cactus, where finite differencing is performed using a regular grid, the number of neighbors is determined by the dimensionality of the problem and the stencil size. Profiling a greater number of applications would of course improve the coverage of this study, but the eight applications studied here broadly represent a wide range of scientific disciplines and modern parallel algorithms.

We also note that in order to study steady-state communication characteristics, we use IPM's regioning feature, which allows us to examine only the profiling data from one section of the code. In particular, we eliminate the large amounts of communication caused by applications during initialization, primarily consisting of large transfers of input data from one node to all of the others.

## 4. COMMUNICATION CHARACTERISTICS

In this section, we analyze the communication characteristics of the eight scientific codes in our study, using the IPM profiling layer to quantify four machine-independent communication properties. First, we identify the type and frequency of application-issued MPI calls, and show the range of buffer sizes utilized for both point-to-point and collective communications by each application. We then profile the communication topology and TDC of each application. These characteristics are further analyzed in the next section.

### 4.1 Call Counts

Table 3 shows the breakdown of the major MPI communication and synchronization calls used by each of our studied applications. Overall, a small subset of calls accounts for most of the observed communication; in most cases, five or fewer MPI calls account for 90% or more of the total communication calls for a particular application. Most codes primarily utilize (over 90% of all MPI calls) point-to-point communication functions, except in the case of GTC, which relies heavily on `MPI_Gather`, and MADbench, which uses `MPI_Reduce` and `MPI_Bcast`. Non-blocking communication (using `Isend`/`Irecv`/etc) is the predominant point-to-point communication model for these codes, although MADbench uses mostly blocking Sends and Receives.

### 4.2 Collectives Buffer Size

Figure 1 shows the cumulatively histogrammed buffer sizes for collective communication for each code. Most of our applications predominantly use relatively small buffer sizes for collective communication. In most cases, the overwhelming majority of buffer sizes are less than 2 KB. However, there is one notable exception: MADbench, which uses very large collective buffer sizes; the preponderance of collective messages for this application use buffer sizes larger than 100 KB. In effect, this code effectively utilizes the interconnect capacity even for collective messages. Overall we see that

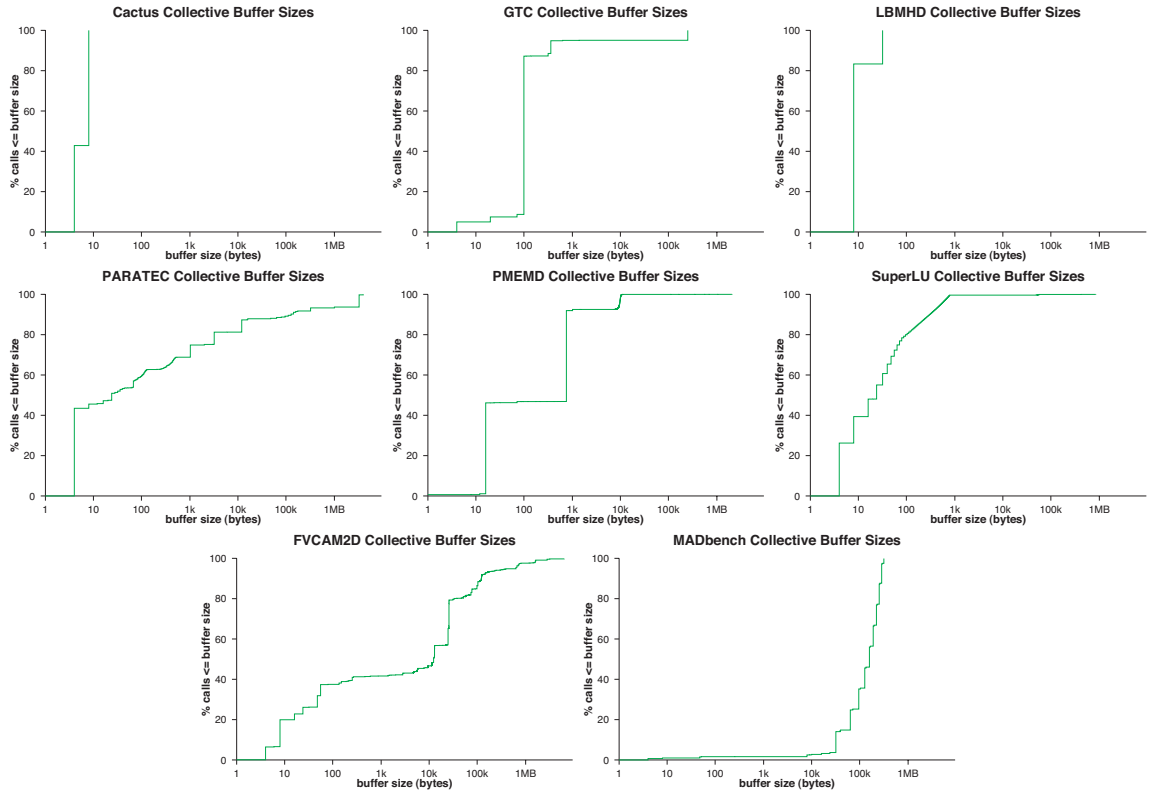



**Figure 1: Buffer sizes distribution for collective communication for all codes.**

with one exception, collective communication tends to occur with buffers smaller than the typical bandwidth-latency product of current and bleeding-edge interconnects.

## 4.3 Point-to-Point Buffer Size

The cumulatively histogrammed buffer sizes for point-to-point communication are shown in Figure 2. Observe that Cactus, GTC, and LBMHD use a relatively small number of buffer sizes. For all three of these codes, most point-to-point communication uses buffer sizes of 100 KB or larger; in the case of LBMHD, almost all buffers are 1 MB or larger. In contrast, the other codes use a large variety of buffer sizes in their point-to-point communication. PARATEC's communication uses a large number of small messages, but the other codes use many large messages of size greater than 10 KB. SuperLU and PMEMD tend to have many small messages as well as large ones, but the other codes, including FVCAM and MADbench, predominantly use large buffers of 10 KB or more. Across all applications, we observe a large mix of buffer sizes, with some applications using a few sizes and others using a large variety of different buffer sizes. Note that in most cases, many of the point-to-point messages are larger than the 2 KB bandwidth-delay product demarcated by the red line; besides the small messages used by SuperLU and PARATEC, the other applications do utilize the available bandwidth in their point-to-point messaging.

## 4.4 Connectivity

In this section, we present the topological connectivity for each application by showing the volume of message exchange between all tasks. Using IPM to record statistics about these message exchanges, we then use the IPM data to construct an undirected graph that represents the topological connectivity of the application. From this graph we calculate certain reduced quantities that describe the communication pattern at a coarse level. Such reduced metrics are important because they allow us to make direct comparisons between applications. In particular, we derive the maximum and average TDC (connectivity) of each code which a key metric for evaluating different interconnect approaches as well by quantifying *how much* connectivity each application requires. We also use a thresholding heuristic based on the bandwidth-delay product (see Section 2.1) that disregards smaller latency-bound messages, sometimes lowering the average and maximum TDC substantially. In this manner, we limit *which* messages we believe contribute most to interconnect utilization and use these messages in our analysis.

Figure 3 shows the regular communication structure exhibited by GTC. This particle-in-cell calculation uses a one-dimensional domain decomposition across the toroidal computational grid, causing each processor to exchange data with its two neighbors as particles cross the left and right boundaries. Additionally, there is a particle decomposition within each toroidal partition, resulting in an average TDC of 4 with a maximum of 17 for the $P = 256$ test case. This maximum TDC is further reduced to 10 when using our 2 KB bandwidth-delay product message size cutoff. These small TDC requirements indicate that most links on an FCN are not being utilized for the GTC simulation.

In the topology chart in Figure 4, we see that the ghost-zone exchanges of Cactus result in communications with "neighboring" nodes, represented by diagonal bands. In fact, each node communicates with at most 6 neighbors due

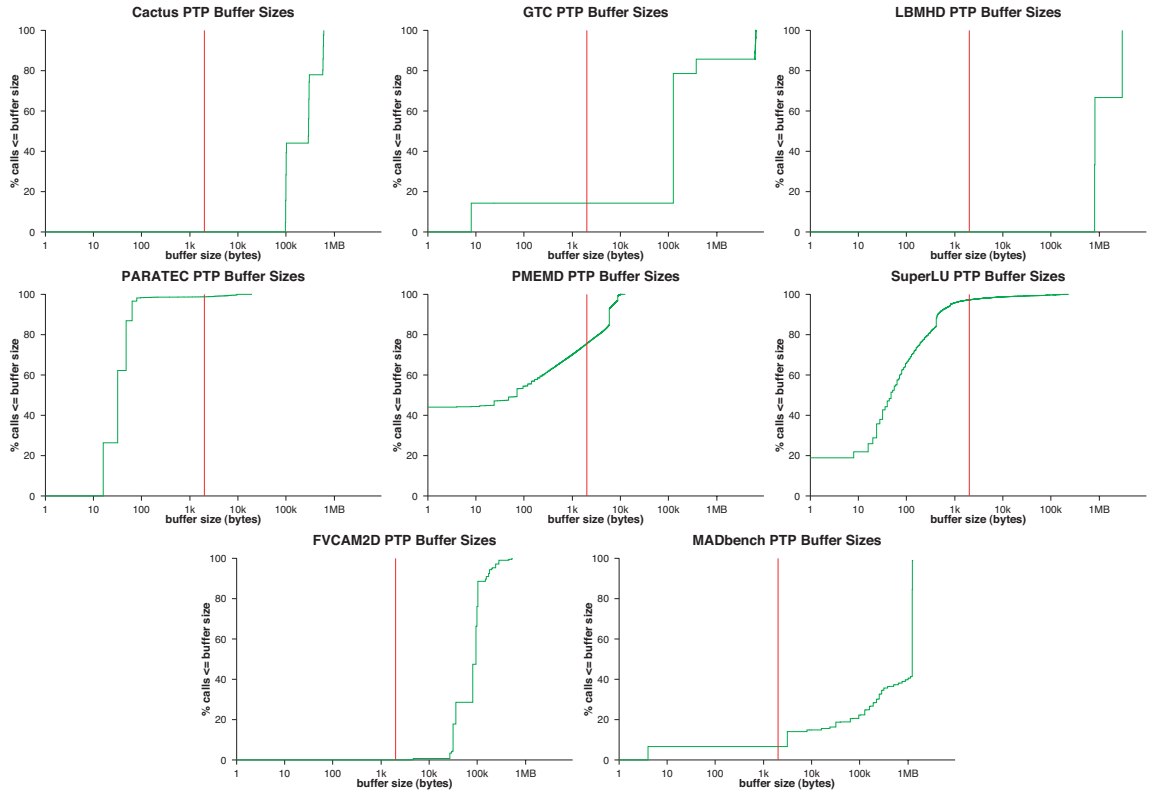



**Figure 2: Buffer sizes distribution for point-to-point communication. The red line demarcates the bandwidth-delay product.**

to the regular computational structure of this 3D stencil code. On average, the TDC is 5, because some nodes are on the boundary and therefore have fewer communicating partners. The maximum TDC is independent of run size and is insensitive to thresholding, but the low TDC indicates limited utilization of an FCN architecture.

The connectivity of LBMHD is shown in Figure 5 for both the 3D and 2D versions. Structurally, we see that the communication, for the 3D version, unlike Cactus, is scattered (not occuring on the diagonal). This is due to the interpolation between the diagonal streaming lattice and underlying structure grid. Note that although the 3D LBMHD streams the data in 27 directions, the code is optimized to reduce the number of communicating neighbors to 12. This degree of connectivity is insensitive to the concurrency level. In contrast, the 2D version of LBMHD has a lower connectivity, on average, of 4. This is due to the lower number of directions that each node communicates for the 2D decomposition. For both versions of LBMHD, the application is insensitive to thresholding for the bandwidth-delay product. In either case, we see that the vast majority of the links in an FCN remain unutilized by this application.

Figure 6 shows the connectivity for our SuperLU run. The complex communication structure of this computation results in many point-to-point message transmissions: in fact, without thresholding the maximum connectivity is equal to $P$. However, by removing the latency-bound messages by thresholding at 2 KB, the average and maximum TDC is reduced to 30 for the 256 processor test case. Also, note that the average connectivity of SuperLU is a function of concurrency, scaling proportionally to $\sqrt{P}$. SuperLU–DIST communication is described more completely in a 2003 paper by Li and Demmel [11].

Figure 7 shows the complex structure of communication in the PMEMD particle mesh ewald calculation. Here the maximum and average TDC is equal to $P$. For the spatial decomposition used in this algorithm, each task's data transfer with another task drops off as their spatial regions become more distant. The rate of this drop off depends strongly on the molecule(s) in the simulation. For $P = 256$, thresholding at 2 KB reduces the *average* connectivity to 55, even though the maximum TDC remains at 256.

In Figure 8 we see the communication requirements of PARATEC. This communication-intensive code relies on global data transposes during its 3D FFT calculations, resulting in large global message traffic [8]. Here the maximum and average TDC is equal to $P$, and the connectivity is insensitive to thresholding. Thus, PARATEC represents the class of codes that can make use of the high bisection bandwidth that a fully-connected network configuration can provide. This type of communication presents formidable challenges to any low-degree connectivity solution.

The communication topologies for FVCAM and CAM are shown in Figure 9. Note that because this application uses OpenMP for every four processors along with MPI, we represent each 4-processor OpenMP task as a single node. For the 1D version (CAM) we see that, as expected, each node communicates with its two neighbors in 1D. Thus, the maximum and average TDC is 2. FVCAM, however, shows a more complex communication structure. Along with the communication close to the diagonal, we see "bands" of communication of lesser total data. Some of the nodes communicate

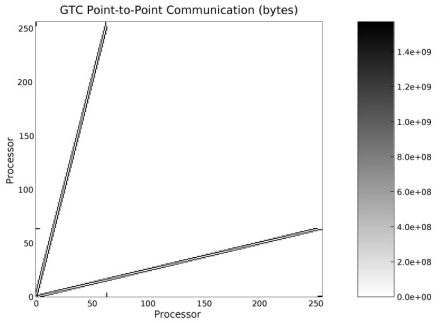



**Figure 3: Communication topology of GTC.**

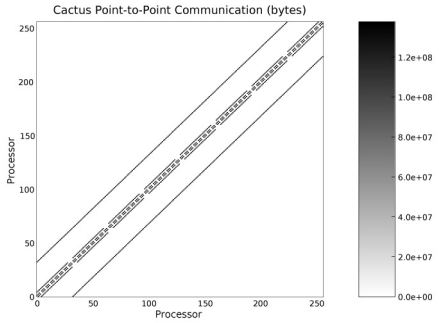
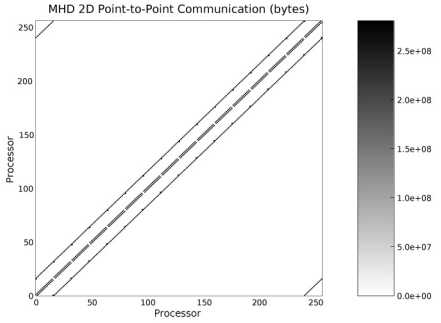



**Figure 4: Communication topology of Cactus.**



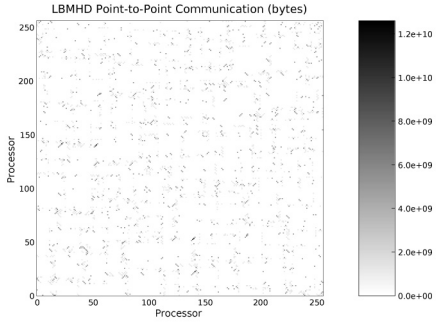




**Figure 5: Communication topology of LBMHD2D (l) and LBMHD3D (r).**

with as many as 20 others, but the average TDC for FVCAM is 15. Neither CAM nor FVCAM is sensitive to our bandwidth-delay thresholding heuristic. However, in both cases, an FCN remains underutilized due to the low TDC. These two graphs demonstrate that different decompositions of the same code can result in different communication.

Lastly, we present the communication topology of MADbench in Figure 10 for two different scheduling schemes. MADbench MG uses ScaLAPACK's `pdgemr2d` function to remap subsets of the matrices to different processors before calling `pdgemm`. The version without gang-scheduling calls `pdgemm` without remapping. As seen in Figure 10, the two versions result in very different communication. The topology of MADbench MG displays the communication required for remapping the matrices, while the graph on the right shows no such characteristics. Without the matrix remapping, MADbench SG has an average TDC of 13, with a max of 15. The version of MADbench that calls `pdgemr2d` has a higher maximum TDC of 44, with an average of 40. Thresholding has no effect in reducing the TDC of either version. Nevertheless, neither version utilizes a large percentage of the available links in an FCN.

Table 4 presents a summary of the application communication characteristics derived in this section. (SuperLU and PMEMD exhibit misleadingly low median point-to-point buffer sizes, but this is due to the fact they sometimes send with buffer sizes of 0 bytes, in cases where a communicating partner expects a message that is not necessary for the computation.)

In the next section, we utilize these results to determine the applicability of different interconnect architectures to each of the applications in this study.

# 5. ANALYSIS

Based on the information gathered in Section 4, we now analyze the communication data in the context of contemporary switch architectures.

One important simplifying assumption we apply to our analysis is that each node contains only a single processor. While most practical systems will likely use SMP nodes, the analysis would need to consider bandwidth localization algorithms for assigning processes to nodes in addition to the analysis of the interconnection network requirements. However, bandwidth localization is a separable issue — one that unnecessarily complicates our analysis of the interconnect behavior and requirements. Therefore, we focus exclusively on single-processor nodes in this paper, and leave the analysis of SMP nodes for future work.

## 5.1 Collectives

Consistent with previous research [18], Figure 1 shows that for many applications, nearly all of the collective communication payload sizes fall below 2 KB. MADbench, however, utilizes many large collective messages, and would therefore benefit from higher-bandwidth interconnects rather than focusing on minimizing latency. Overall, the collective messages are dominated by very small message payload sizes, which are primarily latency bound. Therefore, the collective message requirements of these applications benefit more from latency minimization strategies than they do from higher bandwidth.

The collective communication latency requirements are clearly differentiated from those of the point-to-point messages, which are more firmly bandwidth bound. The typical approach to collective communication is to overlay a minimum-latency routing pattern over the primary high-

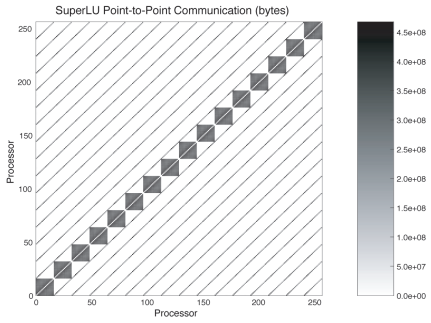



**Figure 6: Communication topology of SuperLU.**

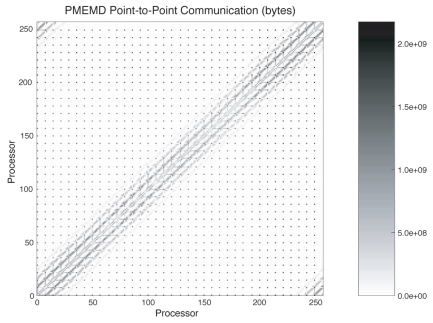



**Figure 7: Communication topology of PMEMD.**

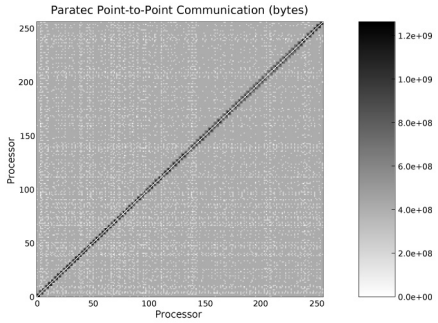



**Figure 8: Communication topology of PARATEC.**

bandwidth interconnect topology, but the latencies of different paths through the interconnect can be dramatically affected by the topology of the underlying communication infrastructure. The complexity of such overlay routing becomes highly dependent on message payload size and job placement.

The IBM Blue Gene/L uses an alternative approach by providing a separate dedicated network with a tree topology that better matches collective communication patterns. Providing a separate dedicated interconnect to support collectives can offer significant performance benefits for Particle-in-Cell (PIC) codes like GTC that rely heavily on collective communications for load-balancing as well as PDE solvers that require fast global reductions to implement implicit methods using Krylov subspace algorithms.

## 5.2 Point-To-Point Traffic

We now discuss each of the applications and consider the class of network best suited for its point-to-point communication requirements. First, we examine the four codes exhibiting the most regularity in their communication exchanges: Cactus, LBMHD, CAM, and GTC. Cactus displays a bounded TDC independent of run size, with a communication topology that isomorphically maps to a regular mesh; thus a fixed 3D mesh/torus would be sufficient to accommodate these types of stencil codes. LBMHD3D also displays a low degree of connectivity, but while its communication pattern is isotropic, the structure is not isomorphic to a regular mesh or torus. The 2D version of this code, however, maps very well to a mesh network, due to the low average TDC of 4, as well as the fact that it is isomorphic to such a network. Although GTC's primary communication pattern is isomorphic to a regular mesh, it has a maximum TDC that is quite higher than the average. For most of the nodes, however, the TDC is low enough to map onto a regular mesh. Lastly, CAM has an average and maximum TDC of 2; this application in fact maps well to a mesh or torus network. Thus, for these four codes, a fixed mesh/torus topology would be well suited for providing sufficient interconnect performance.

SuperLU and PMEMD exhibit anisotropic communication patterns with a TDC that scales with the number of processors. For FVCAM and MADbench, we observe a relatively low TDC, but neither of these codes has an average TDC low enough to map to a torus or regular mesh network. Additionally, PMEMD has widely differing maximum and average TDC. A regular mesh or torus would be inappropriate for this class of computation, but an FCN remains underutilized.

Finally, PARATEC represents the communications requirements for a large class of important chemistry and fluids problems where part of the problem is solved in Fourier space. It requires large global communications involving large messages that fully utilize the FCN. PARATEC's large global communications are a result of the 3D FFTs used in the calculation, which require two stages of global 3D transposes. The first transpose is non-local and involves communicating messages of similar sizes between all the processors, resulting in the uniform background of 32 KB messages. In the second transpose, processors only communicate with neighboring processors, resulting in additional message traffic along the diagonal of the graph. A more detailed description of the communication requirements can be found in [8]. The large global communication requirements can only be effectively provisioned with an FCN network.

We note in the previous section that increasing intercon-

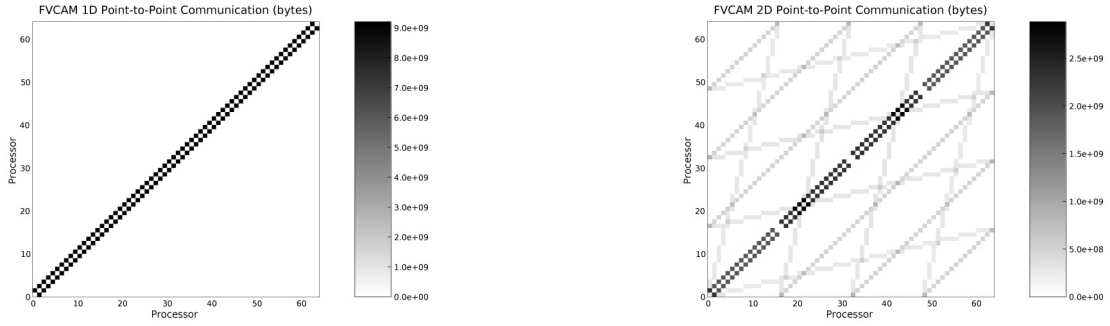



**Figure 9: Communication topology of (l) CAM and (r) FVCAM.**

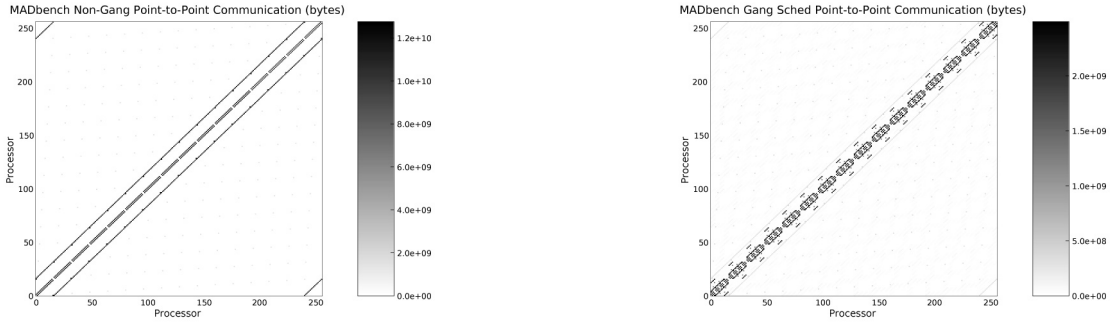



**Figure 10: Communication topology of (l) MADbench SG and (r) MADbench MG.**

nect bandwidth will offer little benefit to the collectives because they are nearly all latency bound. However, Figure 2 shows that the distribution of point-to-point message buffer sizes is predominantly bandwidth bound. One can visualize the effect of adding bandwidth using additional network planes by moving the red vertical bandwidth-delay product line in each graph of Figure 2 to the right. As the bandwidth-delay product increases, the benefits of adding interconnect planes will diminish as a larger fraction of the messages become latency bound. We conclude that increasing interconnect bandwidth will improve performance for the majority of the codes we studied in this project, but the benefits are not directly proportional to the bandwidth increases because of the non-uniform distribution of buffer sizes. We also observe that the distribution of buffer sizes can also exhibit considerable variation depending on problem size and concurrency.

In summary, only two of the eight codes studied exhibit communication patterns that map isomorphically to a 3D mesh network topology. Only one of the codes, PARATEC, fully utilizes the FCN at large scales. The preponderance of codes do not fully utilize an FCN, nor do they map well to regular mesh or torus networks. Finally, increasing interconnect bandwidth benefits point-to-point message performance for the majority of the codes we studied, but the benefits diminish as more of the messages become latency bound.

# 6. CONCLUSIONS

We have analyzed eight parallel codes that represent a wide array of scientific algorithms and associated communication topologies. We show that the average collective payload size is so small that they are primarily latency-bound and do not benefit from over-provisioning of bandwidth. Increased bandwidth can benefit point-to-point messaging for some of the applications studied in this paper, but many of the applications then quickly become latency-bound.

We have also shown that the majority of these codes have communication requirements that underutilize the capabilities of a fully connected network. The majority of applications have very low topological degree of connectivity and commensurate bisection bandwidth requirements. However, the communication topology of many of these same applications do not map isomorphically to a low-degree network like a mesh or torus. Optimal mapping of communication topologies to low-degree fixed-topology interconnects (like the 3D torus) has proven challenging even for a low topological degree [5]. Mesh/torus interconnects include many other challenges, such as limited fault tolerance and difficulties involved in batch scheduling for diverse workloads. These deficiencies can be addressed using hybrid circuit switched networks that employ circuit switches to dynamically customize the links in a low-degree interconnect topology to match application requirements and to route around failures. [17]. However, these approaches depend on applications exhibiting low bisection bandwidth requirements. Codes such as PARATEC that require large bisection bandwidth also pose additional problems for lower-degree interconnects.

This is not to say that there do not exist avenues for making lower-degree interconnects successful for HPC applications. At the start of this paper, we assumed that messages that are smaller than the bandwidth-delay product cannot be pipelined. This is a reasonable assumption for MPI codes given existing performance data regarding the software overhead of sending MPI messages [4]. However, if a lower-overhead messaging layer were utilized, then

| Code | Procs | % PTP Calls | median PTP buffer | % Col. calls | median Col. buffer | TDC @ 2KB cutoff(max,avg) | FCN Circuit Utilization (avg.) |
|---|---|---|---|---|---|---|---|
| GTC | 64 | 42.0 | 128k | 58.0 | 100 | 2, 2 | 3% |
| | 256 | 40.2 | 128k | 59.8 | 100 | 10, 4 | 2% |
| Cactus | 64 | 99.4 | 299k | 0.6 | 8 | 6, 5 | 8% |
| | 256 | 99.5 | 300k | 0.5 | 8 | 6, 5 | 2% |
| LBMHD3D | 64 | 99.8 | 811k | 0.2 | 8 | 12, 11.5 | 19% |
| | 256 | 99.9 | 848k | 0.1 | 8 | 12, 11.8 | 5% |
| LBMHD2D | 256 | 100 | 12k | 0.0 | 8 | 4, 4 | 2% |
| SuperLU | 64 | 89.8 | 64 | 10.2 | 24 | 14, 14 | 22% |
| | 256 | 92.8 | 48 | 7.2 | 24 | 30, 30 | 25% |
| PMEMD | 64 | 99.1 | 6k | 0.9 | 768 | 63, 63 | 100% |
| | 256 | 98.6 | 72 | 1.4 | 768 | 255, 55 | 22% |
| PARATEC | 64 | 99.5 | 64 | 0.5 | 8 | 63, 63 | 100% |
| | 256 | 99.9 | 64 | 0.1 | 4 | 255, 255 | 100% |
| FVCAM | 64 | 99.5 | 96k | 0.5 | 8 | 20, 15 | 23% |
| CAM | 64 | 99.6 | 359k | 0.4 | 208 | 2, 2 | 3% |
| MADbench MG | 256 | 78 | 1.2M | 22 | 163k | 44, 40 | 16% |
| MADbench SG | 256 | 97.3 | 327k | 2.7 | 163k | 15, 13 | 5% |

**Table 4: Summary of code characteristics for point-to-point (PTP) and collective (Col.) communications. Note that CAM/FVCAM ran on 256 processors, but only used 64 MPI processes each utilizing 4 OpenMP threads.**

over-provisioning of interconnect bandwidth might mitigate the effects of non-optimal mapping between the application communication topology and that of the underlying interconnect. Furthermore, recent papers such as one by the ARCMI Project [9] indicate that low-overhead messaging can be used to successfully hide bisection bandwidth limitations through more effective overlap of communication and computation. However, the success of these solutions hinge on adoption of runtime environments and programming languages such as UPC, Co-Array Fortran, or Titanium that support low-overhead messaging and underlying hardware mechanisms such as RDMA.

In summary, we have shown that the communication patterns exhibited by these codes generally underutilize the bisection bandwidth of fully-connected networks. Fully-connected networks still remain the most flexible approach to interconnect topology at the moment primarily because of scheduling and job mapping flexibility and *not* due to bisection bandwidth requirements. Over-provisioning bandwidth cannot be used to compensate for the reduced bisection bandwidth of mesh or hybrid circuit switched interconnects unless the overhead for sending small messages can be reduced significantly. We conclude that mesh and hybrid circuit switched interconnects offer compelling alternatives to fully connected networks, but their benefits cannot be realized by hardware alone.

# 7. REFERENCES


[1] Science case for large-scale simulation. In D. Keyes, editor, *DOE Office of Science Workshop*, June 2003.

[2] IPM Homepage. `http://www.nersc.gov/projects/ipm`, 2005.

[3] M. Alcubierre, G. Allen, B. Brgmann, E. Seidel, and W.-M. Suen. Towards an understanding of the stability properties of the 3+1 evolution equations in general relativity. *Phys. Rev. D*, (gr-qc/9908079), 2000.

[4] C. Bell, D. Bonachea, Y. Cote, J. Duell, P. Hargrove, P. Husbands, C. Iancu, M. Welcome, and K. Yelick. An evaluation of current high-performance networks. In *17th International Parallel and Distributed Processing Symposium (IPDPS)*, 2003.

[5] G. Bhanot, A. Gara, P. Heidelberger, E. Lawless, J. C. Sexton, and R. Walkup. Optimizing task layout on the blue gene/l supercomputer. *IBM Journal of Research and Development*, 49(2/3):489–501, 2005.

[6] Julian Borrill, Jonathan Carter, Leonid Oliker, David Skinner, and R. Biswas. Integrated performance monitoring of a cosmology application on leading hec platforms. In *Proceedings of the International Conference on Parallel Processing (ICPP), to appear*, 2005.

[7] A. Canning, J. Carter, J. Shalf, and S. Ethier. Scientific computations on modern parallel vector systems. In *Proceedings of the IEEE Conference on Supercomputing*, 2004.

[8] A. Canning, L.W. Wang, A. Williamson, and A. Zunger. Parallel empirical pseudopotential electronic structure calculations for million atom systems. *J. Comput. Phys.*, 160:29, 2000.

[9] Manoj Krishnan and Jarek Nieplocha. Optimizing performance on linux clusters using advanced communication protocols: Achieving over 10 teraflops on a 8.6 teraflops linpack-rated linux cluster. In *6th International Conference on Linux clusters: The HPC Revolution, Chapel Hill, USA*, 2005.

[10] W. W. Lee. Gyrokinetic particle simulation model. *J. Comp. Phys.*, 72, 1987.

[11] Xiaoye S. Li and James W. Demmel. Superlu–dist: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Trans. Mathematical Software*, 29(2):110–140, June 2003.

[12] Z. Lin, S. Ethier, T.S. Hahm, and W.M. Tang. Size scaling of turbulent transport in magnetically confined plasmas. *Phys. Rev. Lett.*, 88, 2002.

[13] A. Macnab, G. Vahala, P. Pavlo, , L. Vahala, and M. Soe. Lattice boltzmann model for dissipative incompressible MHD. In *Proc. 28th EPS Conference on Controlled Fusion and Plasma Physics*, volume 25A, 2001.

[14] A. Macnab, G. Vahala, and L. Vahala. Lattice boltzmann model for dissipative MHD. In *Proc. 29th EPS Conference on Controlled Fusion and Plasma Physics*, volume 26B, Montreux, Switzerland, June 17-21, 2002.

[15] Leonid Oliker, Jonathan Carter, Michael Wehner, Andrew Canning, Stephane Ethier, B. Govindasamy, A. Mirin, and D. Parks. Leading computational methods on scalar and vector hec platforms. In *Proceedings of Supercomputing 2005, to appear*, 2005.

[16] Rolf Rabenseifner. Automatic profiling of MPI applications with hardware performance counters. In *Proceedings of the 6th European PVM/MPI User's Group Meeting (EuroPVM/MPI)*, pages 35–42, September 1999.

[17] John Shalf, Shoaib Kamil, Leonid Oliker, and David Skinner. Analyzing ultra-scale application communication requirements for a reconfigurable hybrid interconnect. In *Proceedings of Supercomputing 2005 (to appear)*, 2005.

[18] Jeffery S. Vetter and Frank Mueller. Communication characteristics of large-scale scientific applications for contemporary cluster architectures. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS)*, 2002.

[19] Jeffrey S. Vetter and Andy Yoo. An empirical performance evaluation of scalable scientific applications. In *Proceedings of the IEEE Conference on Supercomputing*, 2002.